

# Learning & Verifying Graphs using Queries, with a focus on Edge Counting

Lev Reyzin & Nikhil Srivastava

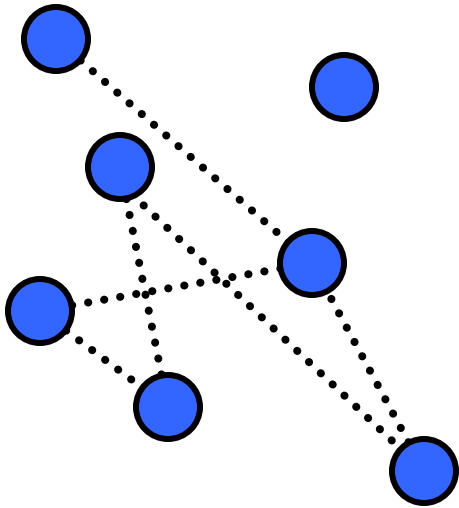
*Yale University*

# Plan

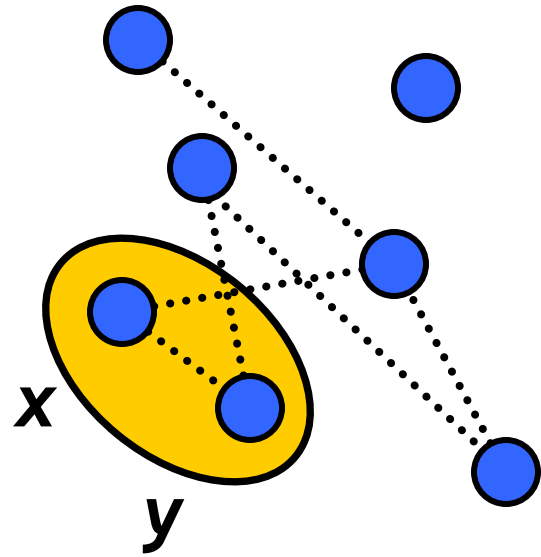
- ED/EC queries
- Learning and Verification Tasks
- Partition in  $O(n \log n)$  EC queries
- Verification w.h.p in  $O(1)$  EC queries
- Open problems



# Queries

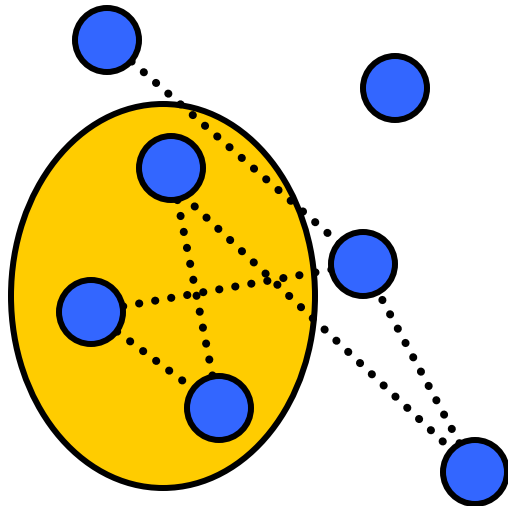


# Queries



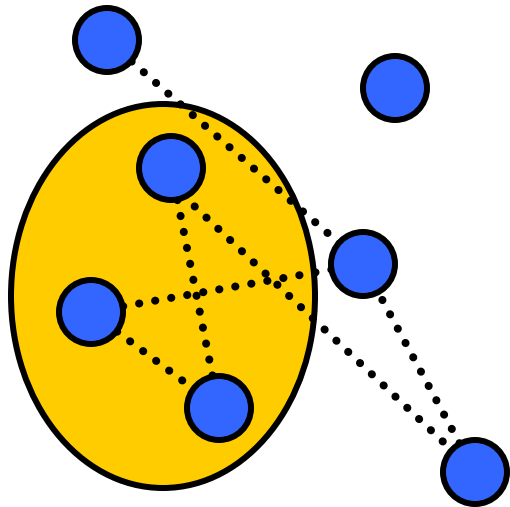
$$1 \{xy \in E\}$$

# Queries

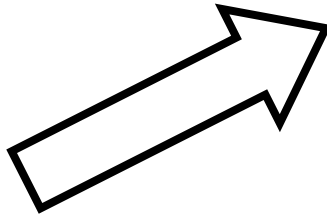


$S \subseteq V$

# Queries



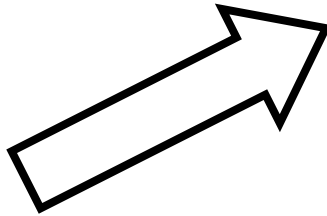
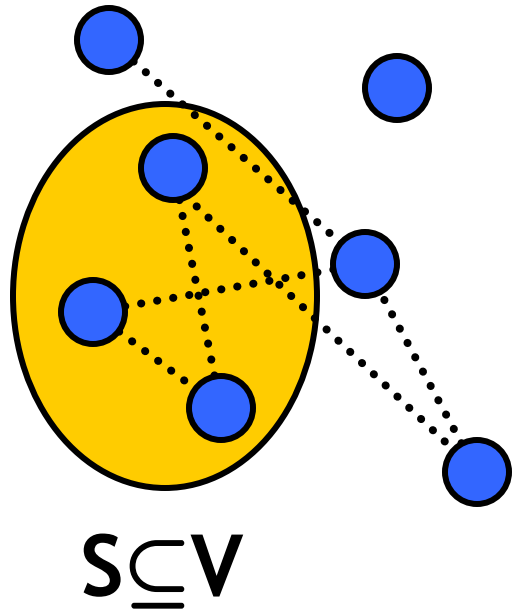
$S \subseteq V$



$ED(S) =$

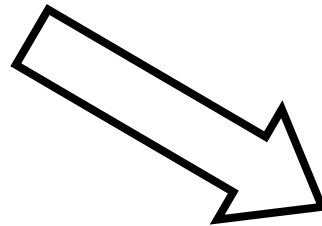
$1_{\{S \text{ contains an edge}\}}$

# Queries



$$ED(S) =$$

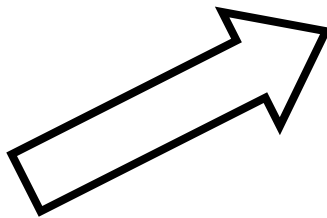
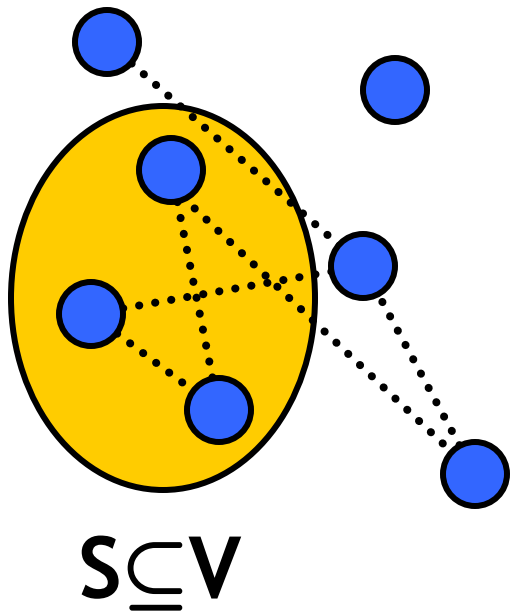
$$1_{\{S \text{ contains an edge}\}}$$



$$EC(S) =$$

(# edges induced by S)

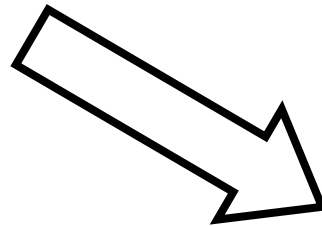
# Queries



$$ED(S) =$$

$$1_{\{S \text{ contains an edge}\}}$$

[AC04,06]



$$EC(S) =$$

(# edges induced by S)

[GK99,00,05]

# Plan

- ED/EC queries
- Learning and Verification Tasks
- Partition in  $O(n \log n)$  EC queries
- Verification w.h.p in  $O(1)$  EC queries
- Open problems

# Learning Tasks

- Learning:
  - Find  $G$ .
  - Find the connected components of  $G$ .
- *Also (not in this work):*  
*girth, chromatic number, diameter,*  
*expansion, edge-connectivity, etc.*
- Verification:
  - Is  $G=G_0$  for some fixed  $G_0$ ?

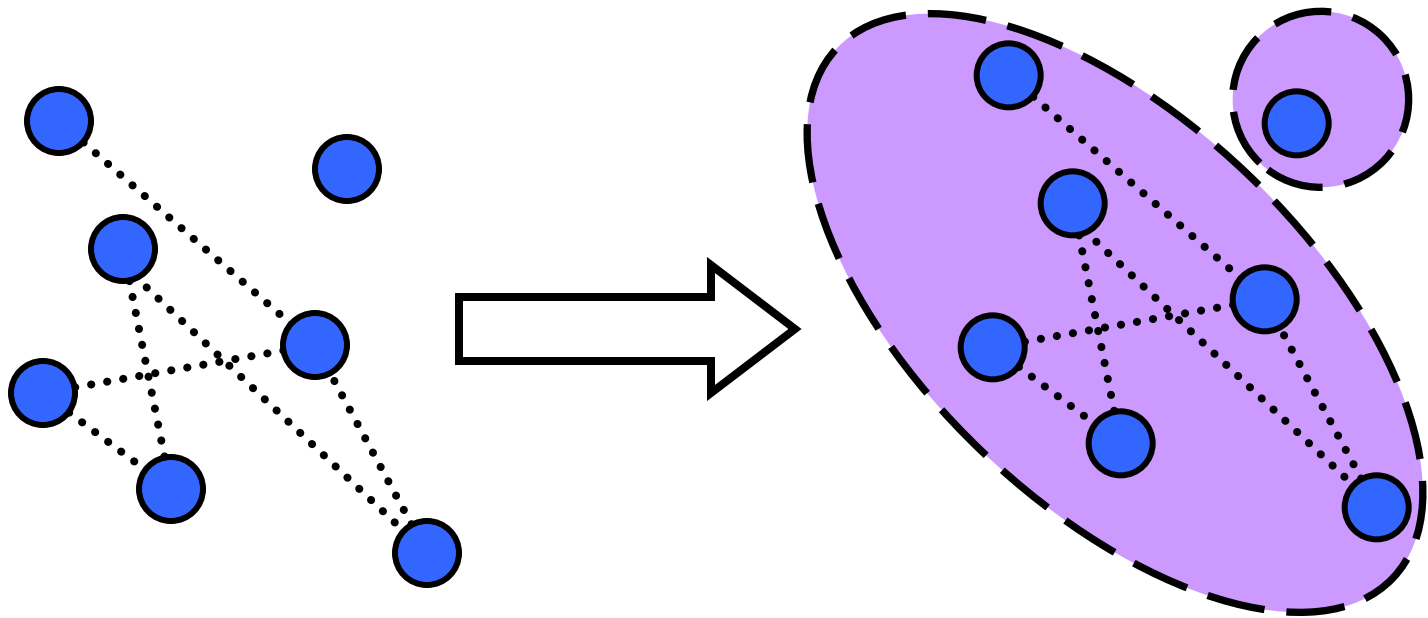
# Motivation

- Natural combinatorial problem
- Chemistry example
  - Vertices of  $H$  are compounds
  - Want to check which pairs 'react'
  - Try subsets  $S$  at a time
- Bioinformatics

# Plan

- ED/EC queries
- Learning and Verification Tasks
- Partition in  $O(n \log n)$  EC queries
- Verification w.h.p in  $O(1)$  EC queries
- Open problems

# Partition



# Partition

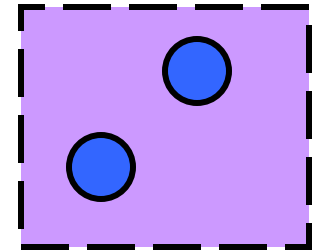
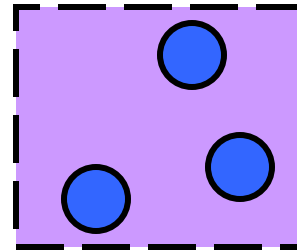
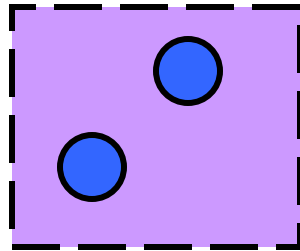
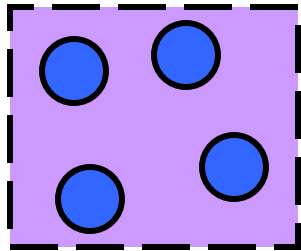
**Thm:** Can learn in  $O(n \log n)$  EC queries.

# Partition

**Thm:** Can learn in  $O(n \log n)$  EC queries:

1. Let  $V = \{v_1 \dots v_n\}$
2. Suppose we know components in  $\{v_1 \dots v_i\}$ .

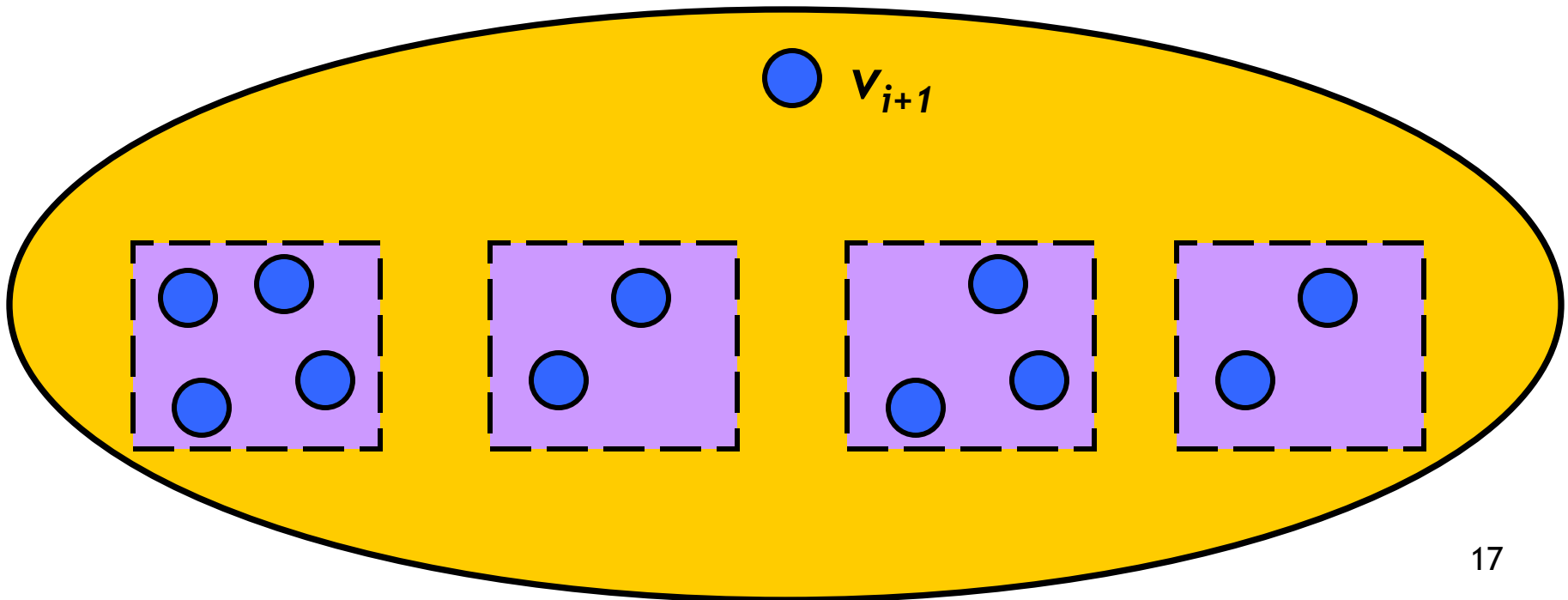
  $v_{i+1}$



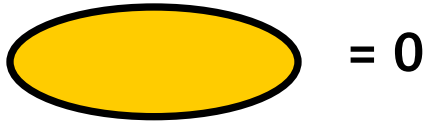
# Partition

**Thm:** Can learn in  $O(n \log n)$  EC queries:

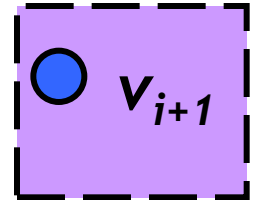
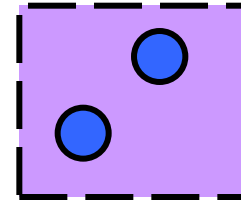
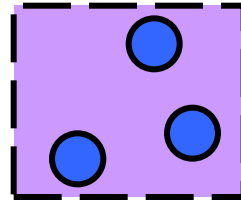
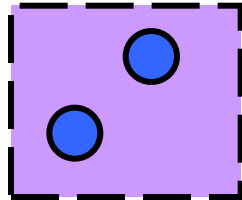
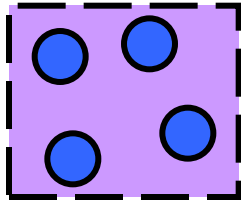
1. Let  $V = \{v_1 \dots v_n\}$
2. Suppose we know components in  $\{v_1 \dots v_i\}$ .



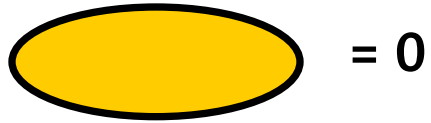
# Partition



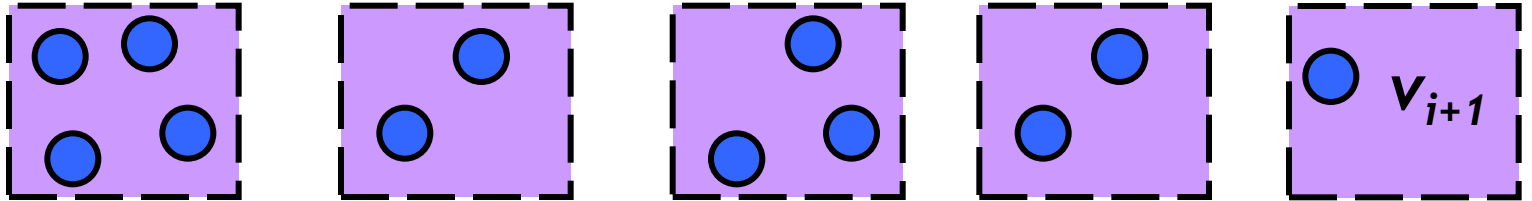
*create new component*



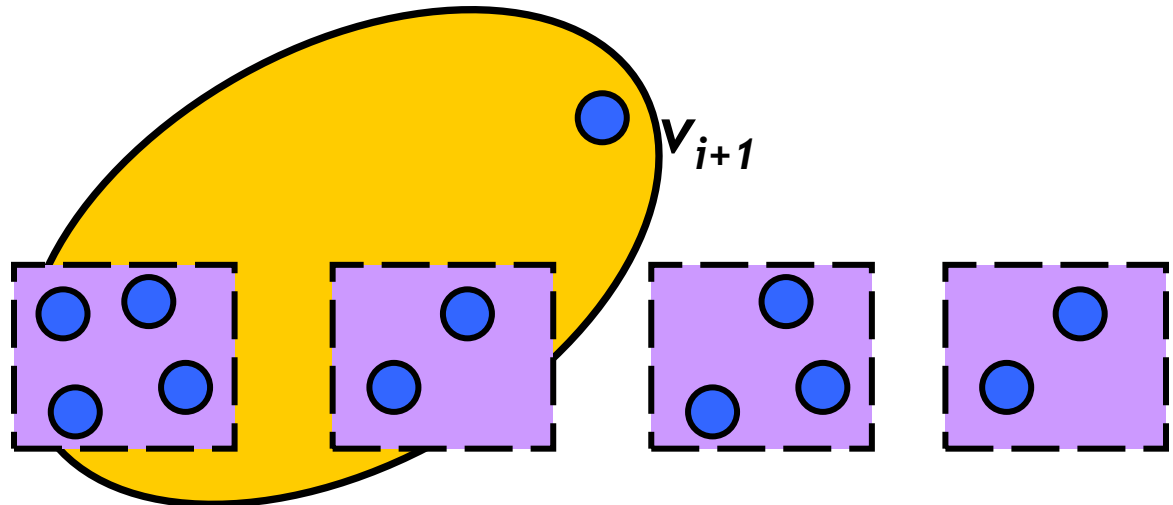
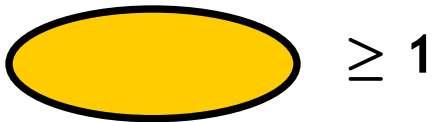
# Partition



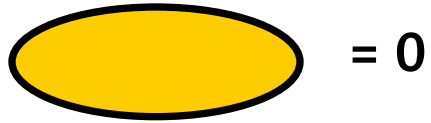
*create new component*



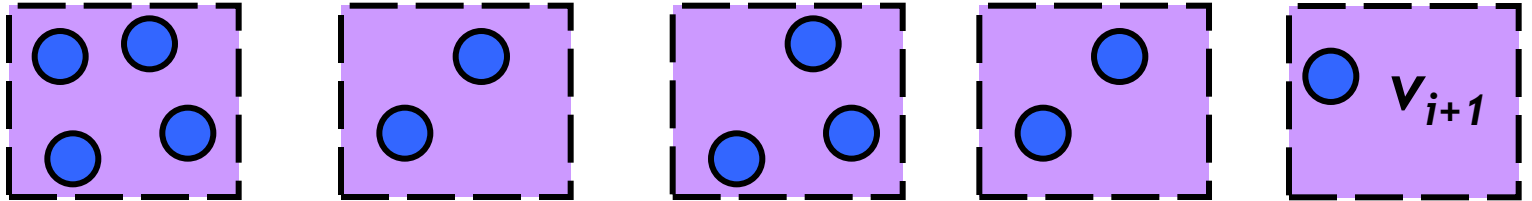
*recurse on half until edge is found*



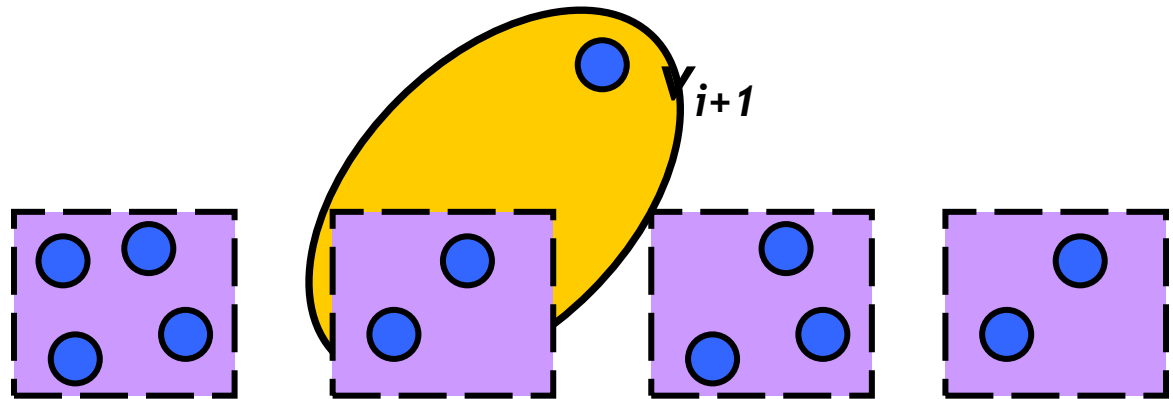
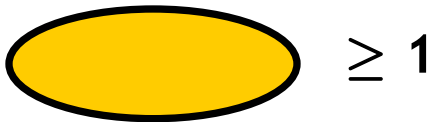
# Partition



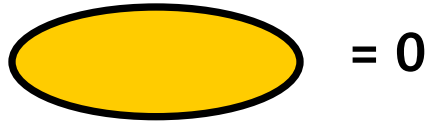
*create new component*



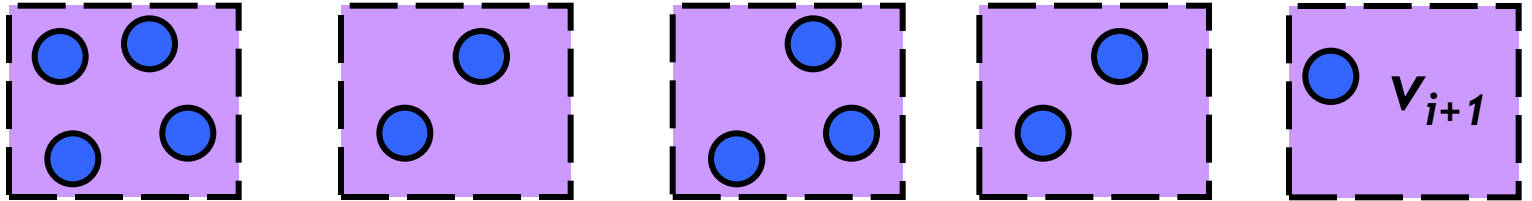
*recurse on half until edge is found*



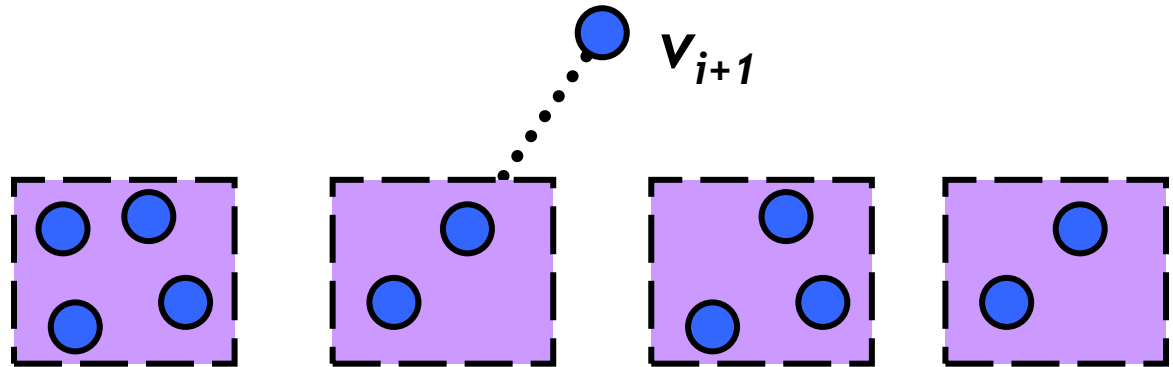
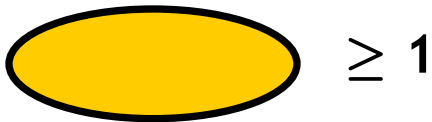
# Partition



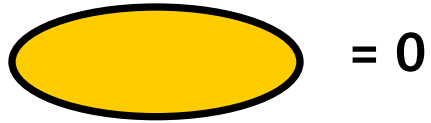
*create new component*



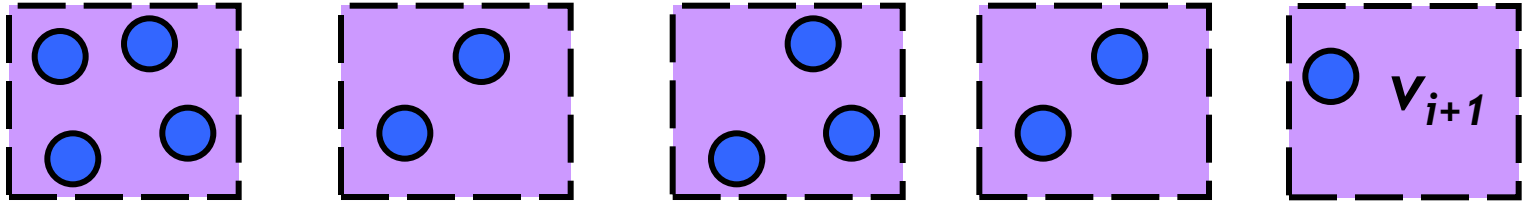
*recurse on half until edge is found*



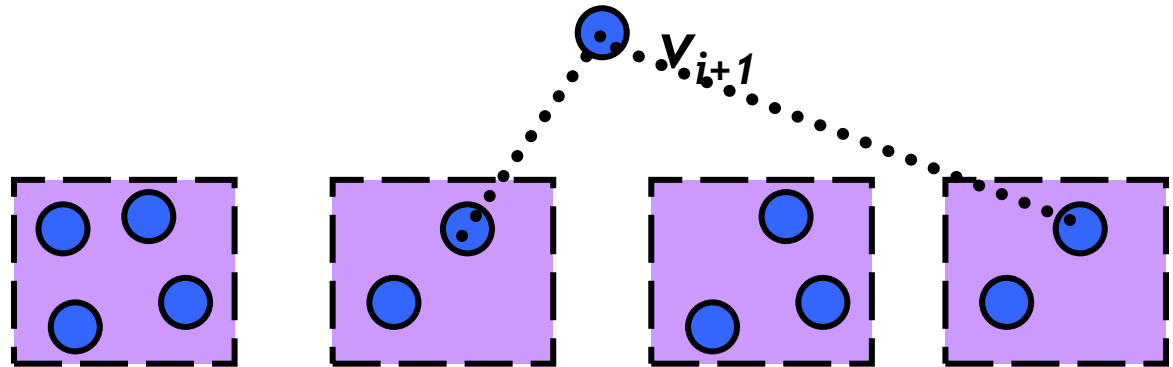
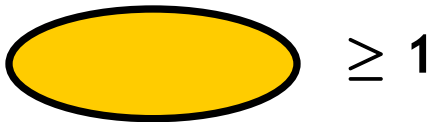
# Partition



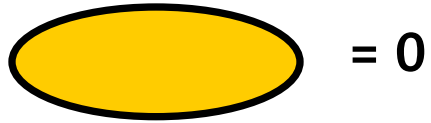
*create new component*



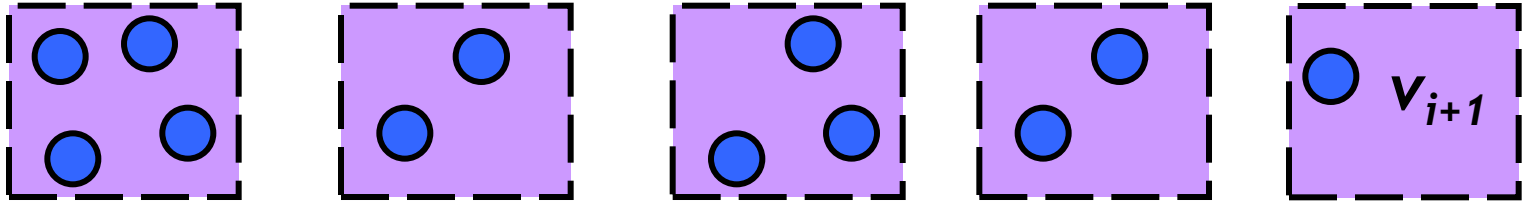
*find one edge to each component*



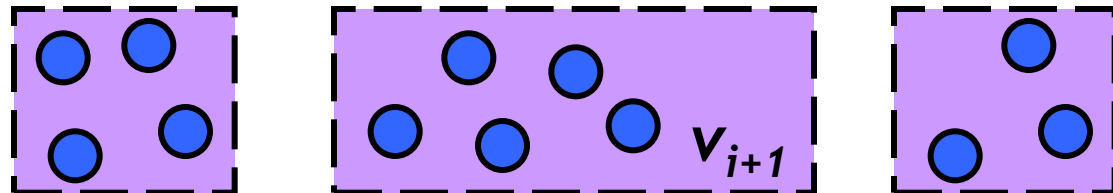
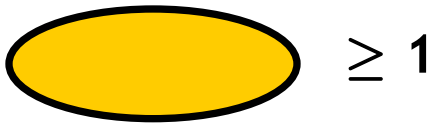
# Partition



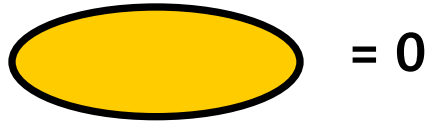
*create new component*



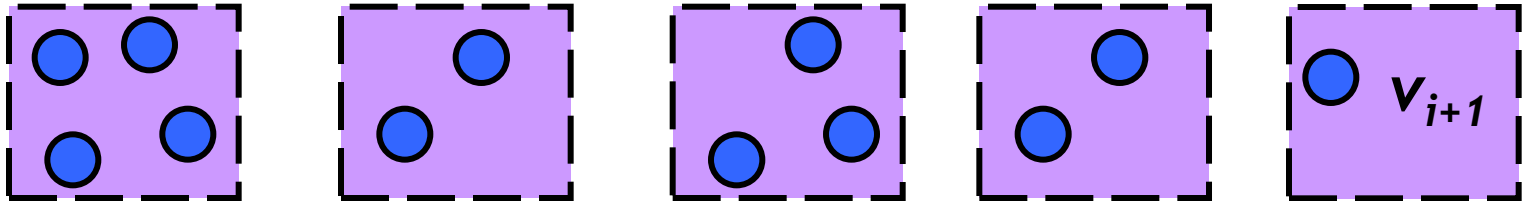
*merge and continue*



# Partition

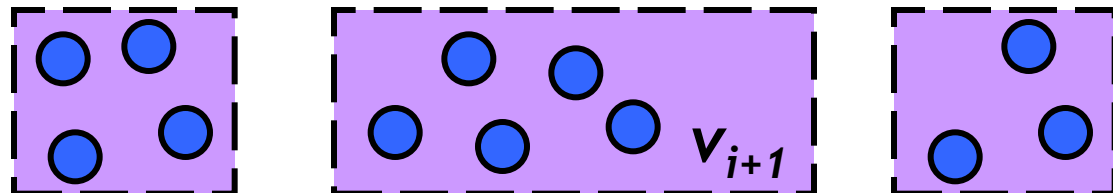
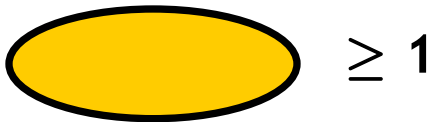


*create new component*

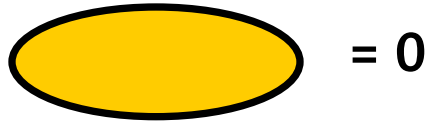


*merge and continue*

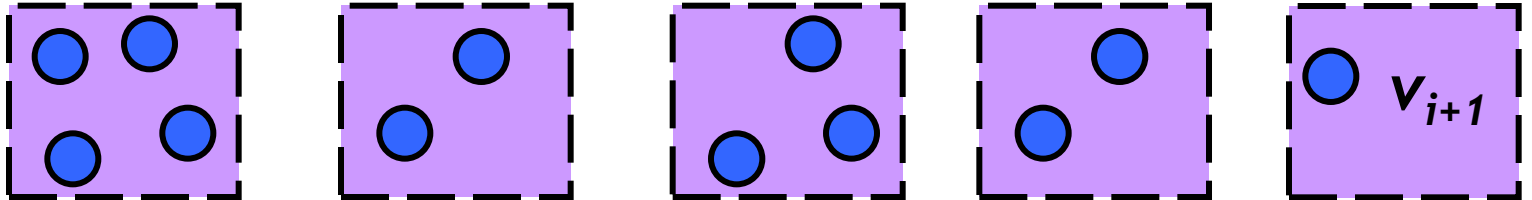
*we never add cycles*



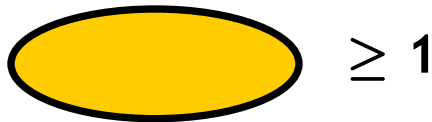
# Partition



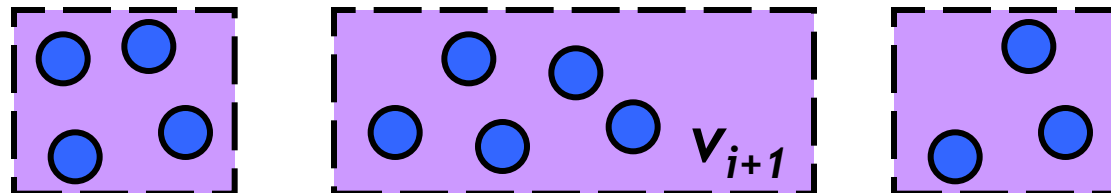
*create new component*



*merge and continue*



*algorithm creates a spanning forest.  
logn queries per added edge.*

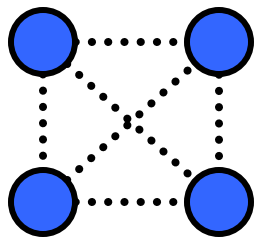
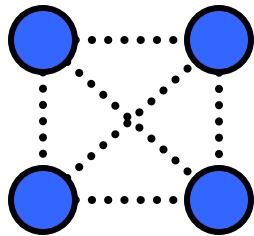


# Plan

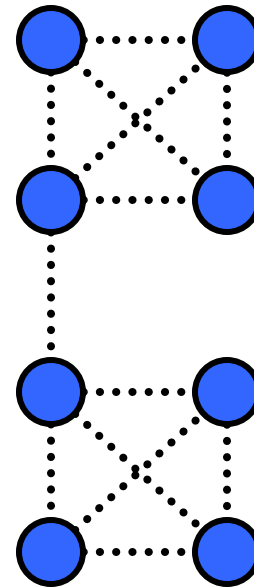
- ED/EC queries
- Learning and Verification Tasks
- Partition in  $O(n \log n)$  EC queries
  - Need  $\Omega(n^2)$  ED queries.
- Verification w.h.p in  $O(1)$  EC queries
- Open problems

# Partition: ED Lowerbound

- $\Omega(n^2)$  ED queries:

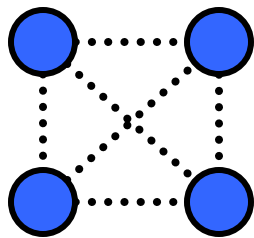
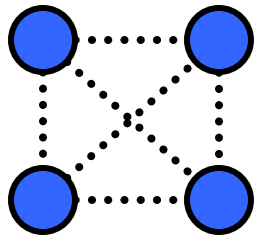


Vs.

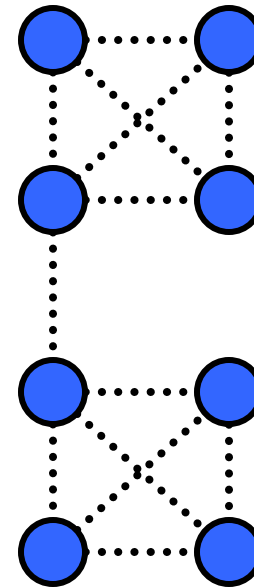


# Partition: ED Lowerbound

- $\Omega(n^2)$  ED queries:



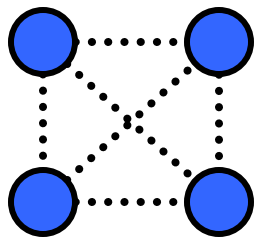
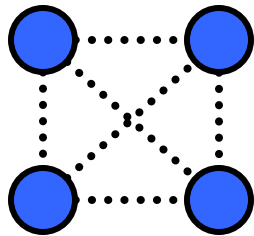
Vs.



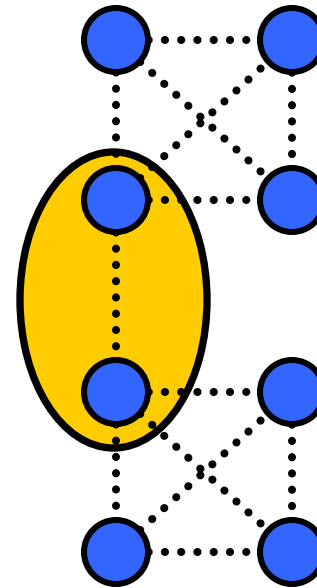
*If  $|S| > 2$ ,  $ED(S) = 1$*

# Partition: ED Lowerbound

- $\Omega(n^2)$  ED queries:



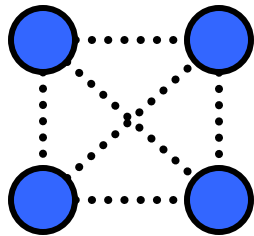
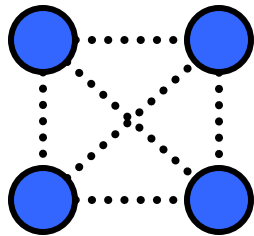
Vs.



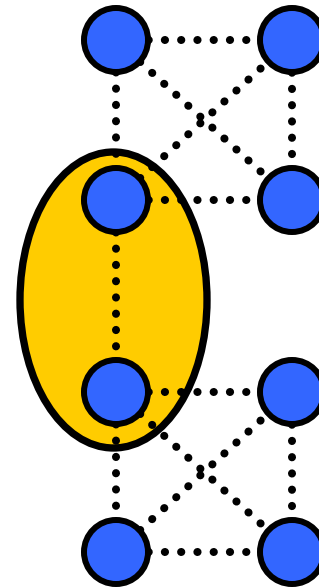
*If  $|S| > 2$ ,  $ED(S) = 1$*

# Partition: ED Lowerbound

- $\Omega(n^2)$  ED queries:



Vs.



*If  $|S| > 2$ ,  $ED(S) = 1$*



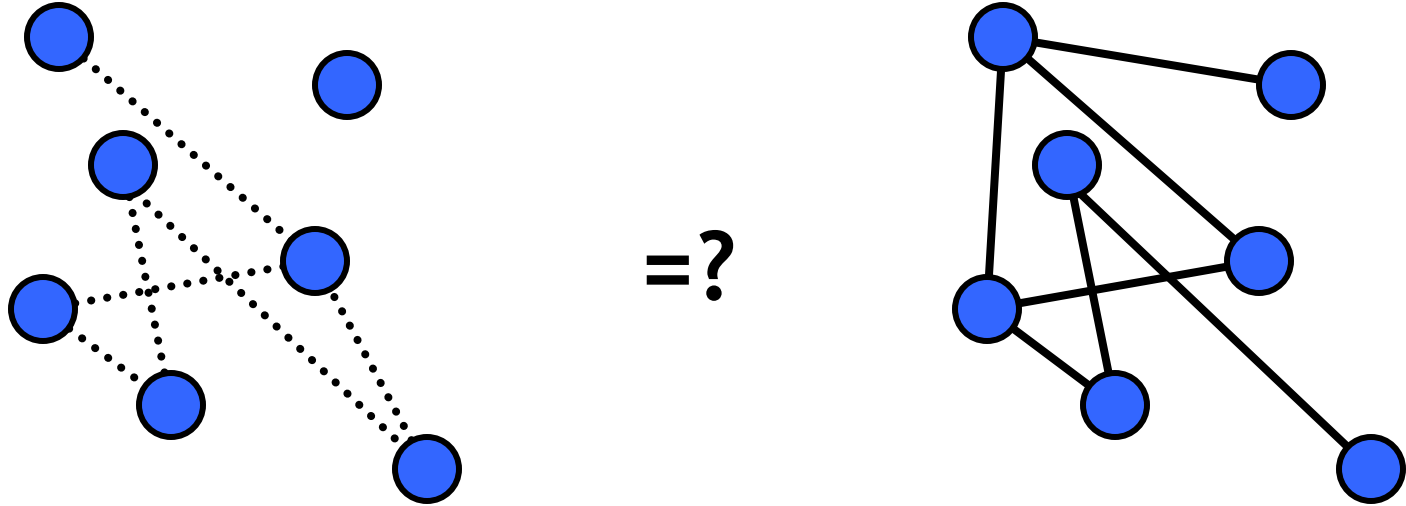
# Partition: EC Lowerbound

- Information theoretic:
  - # partitions  $B_n$
  - $\log B_n = n \log n$
  - Each query gives  $\log(n^2) = 2 \log n$  bits
  - Need  $\Omega(n)$  queries.

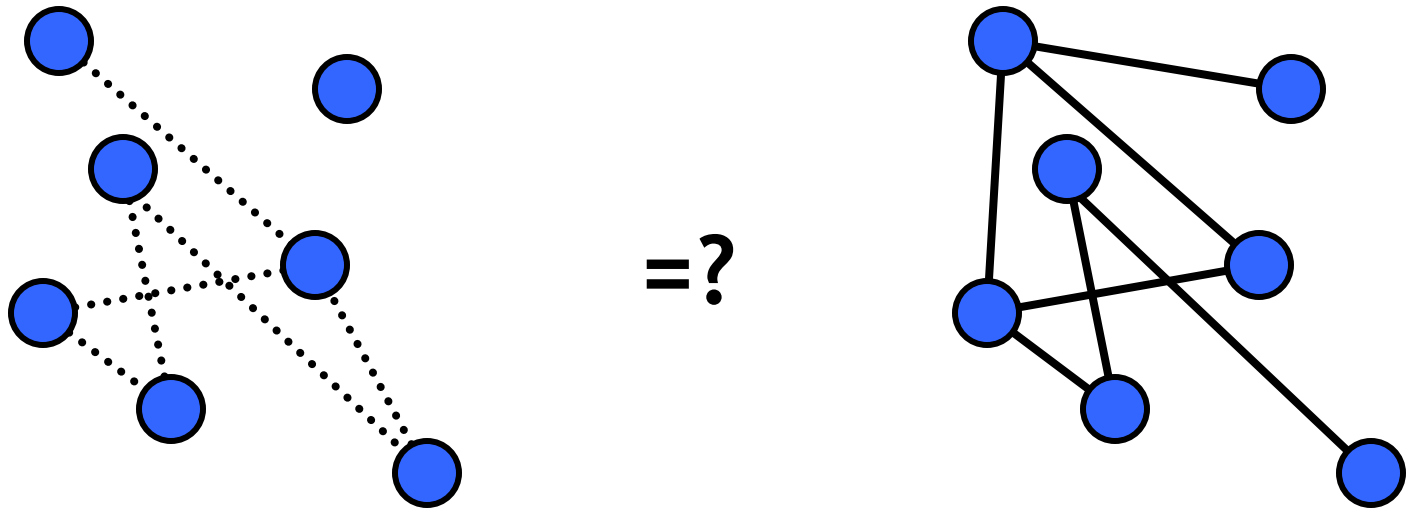
# Plan

- ED/EC queries
- Learning and Verification Tasks
- Partition in  $O(n \log n)$  EC queries
- Verification w.h.p in  $O(1)$  EC queries
- Open problems

# Verification

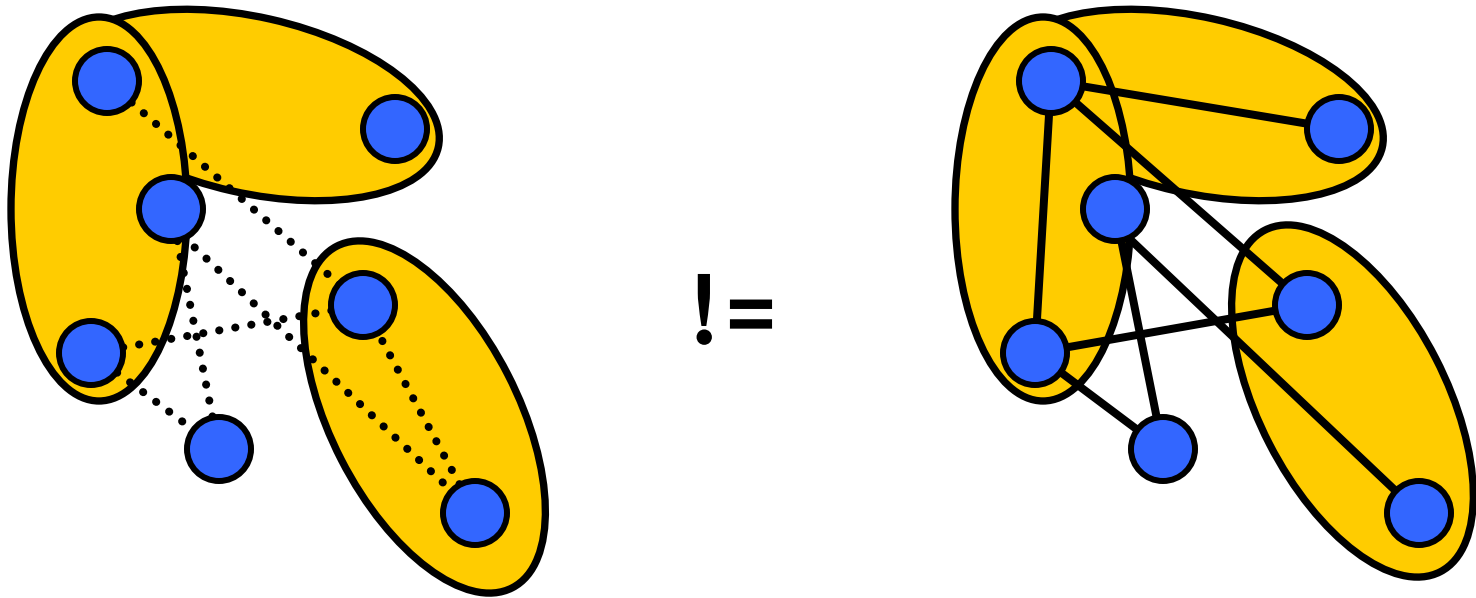


# Verification



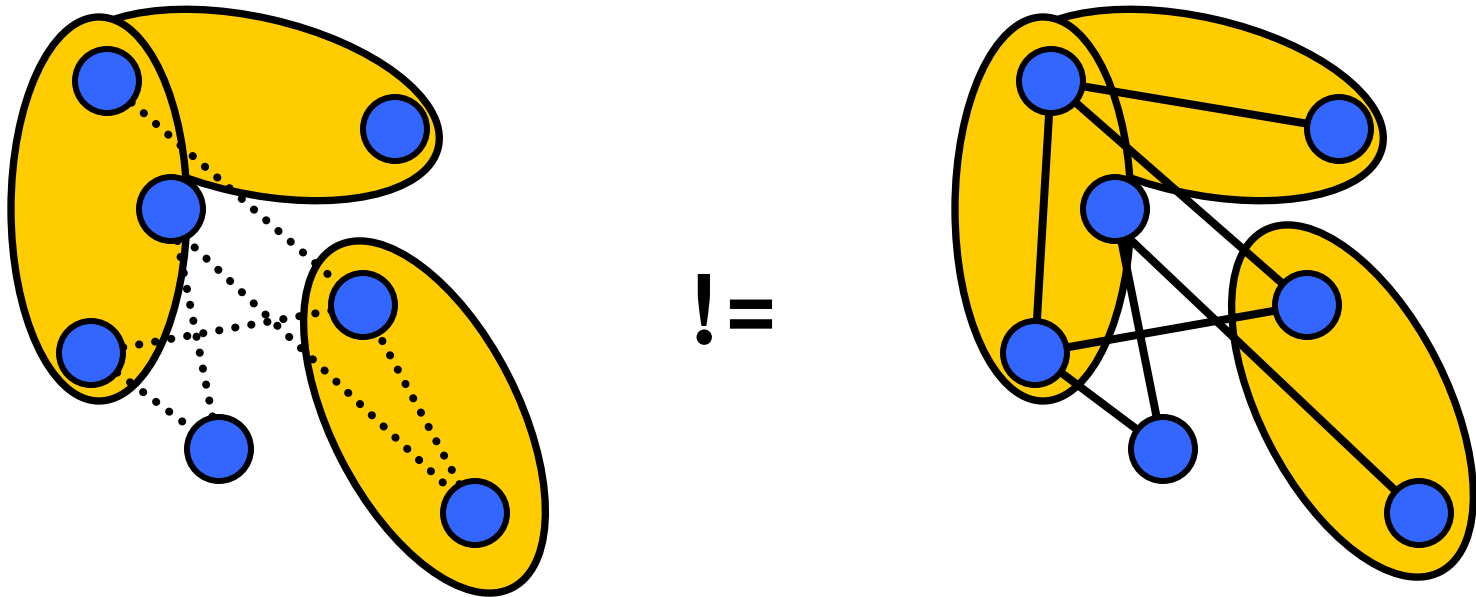
*no harder than learning*

# Verification



*no harder than learning*

# Verification



*motivation: check for errors in learning*

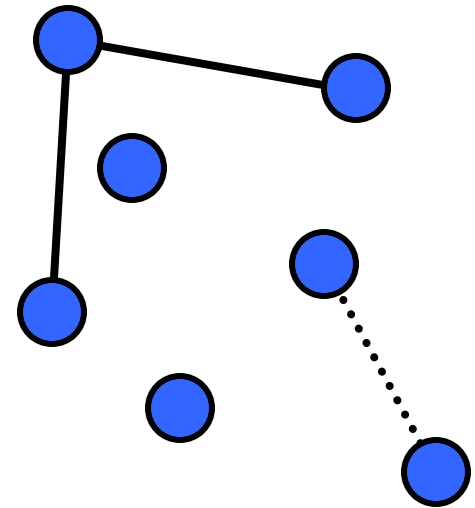
# Verification

**Thm:** If  $G \neq H$  then with probability  $\frac{1}{4}$   
 $EC_G(S) \neq EC_H(S)$  for a random subset  $S$

# Verification

**Thm:** If  $G \neq H$  then with probability  $\frac{1}{4}$   
 $EC_G(S) \neq EC_H(S)$  for a random subset  $S$

**Pf:** If  $G \Delta H \neq \emptyset$ , then with prob.  $\frac{1}{4}$   $G \Delta H(S)$  has  
an **odd** number of edges.

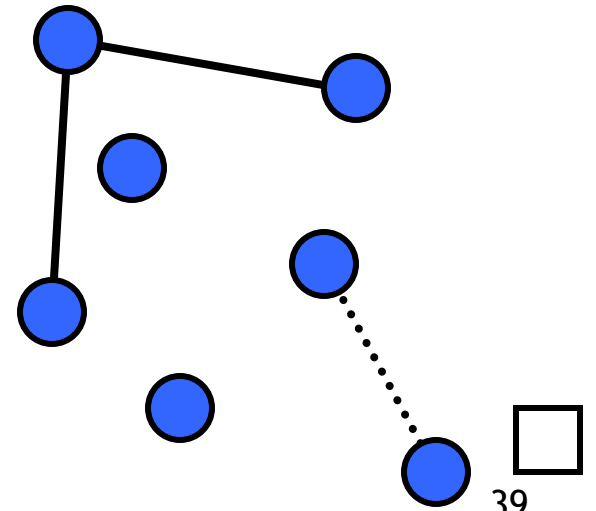


# Verification

**Thm:** If  $G \neq H$  then with probability  $\frac{1}{4}$   
 $EC_G(S) \neq EC_H(S)$  for a random subset  $S$

**Pf:** If  $G \Delta H \neq \emptyset$ , then with prob.  $\frac{1}{4}$   $G \Delta H(S)$  has  
an **odd** number of edges.

So  $EC_G(S) + EC_H(S)$  is odd



# Verification

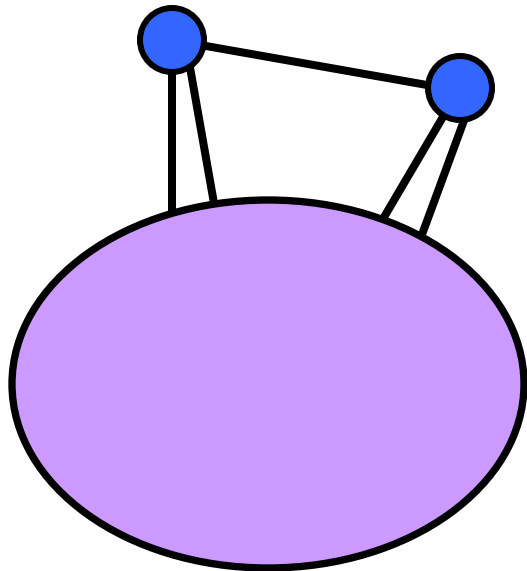
- **Lemma:** If  $G$  is nonempty and  $S$  is a random subset of vertices, then with probability  $\geq \frac{1}{4}$ ,  $S$  induces an **odd** number of edges.

# Verification

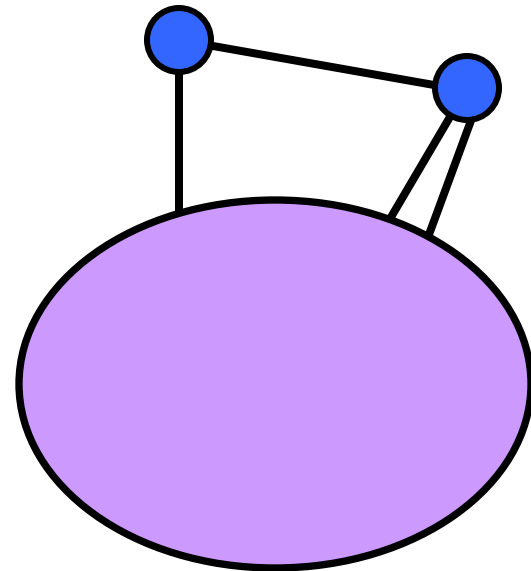
- **Pf of Lemma:** Order the vertices  $\{v_1 \dots v_n\}$  so  $(v_{n-1}, v_n) \in E$ . Choose in order with Pr.  $\frac{1}{2}$ .

# Verification

- **Pf of Lemma:** Order the vertices  $\{v_1 \dots v_n\}$  so  $(v_{n-1}, v_n) \in E$ . Choose in order with Pr.  $\frac{1}{2}$ .

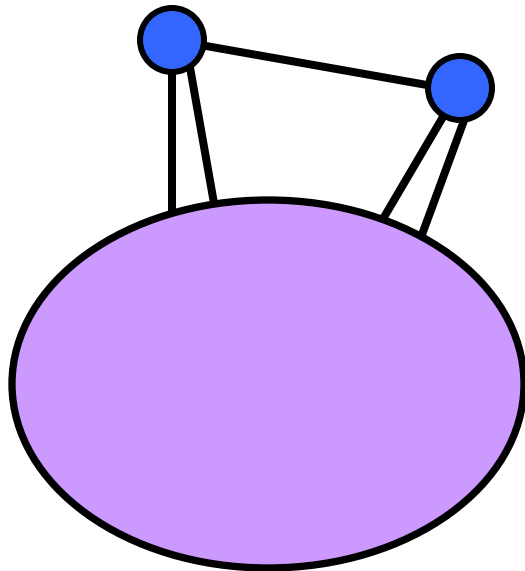


*2 cases*



# Verification

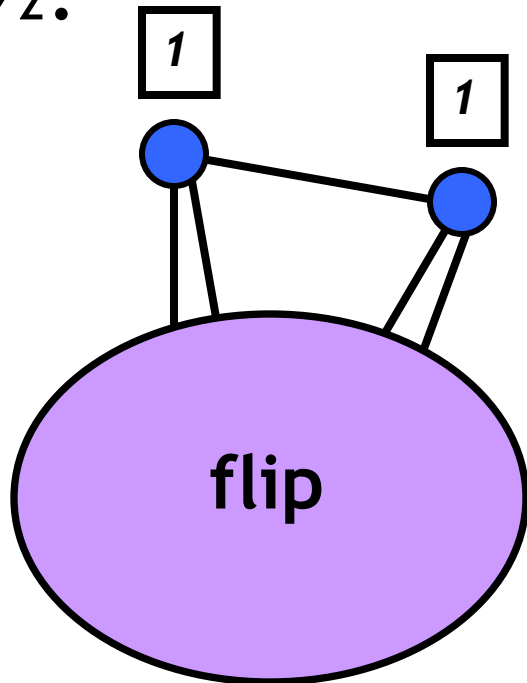
- **Pf of Lemma:** Order the vertices  $\{v_1 \dots v_n\}$  so  $(v_{n-1}, v_n) \in E$ . Choose in order with Pr.  $\frac{1}{2}$ .



*even case*

# Verification

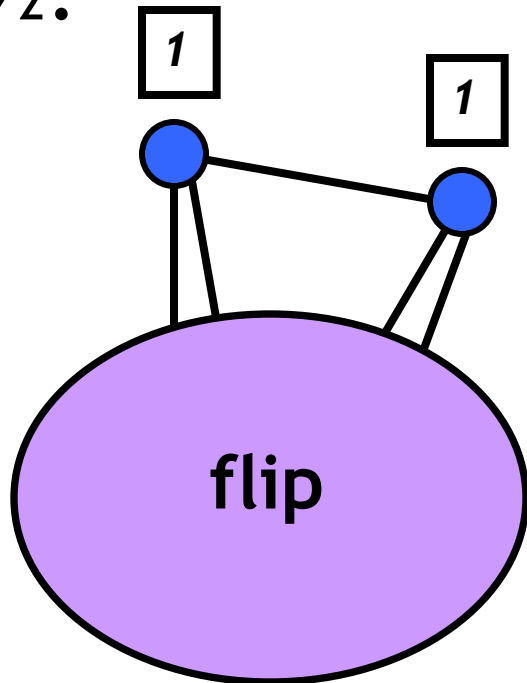
- **Pf of Lemma:** Order the vertices  $\{v_1 \dots v_n\}$  so  $(v_{n-1}, v_n) \in E$ . Choose in order with Pr.  $\frac{1}{2}$ .



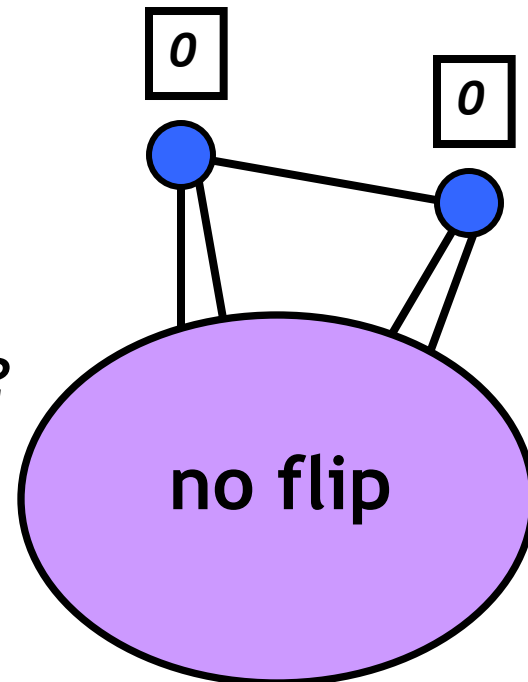
*even case*

# Verification

- **Pf of Lemma:** Order the vertices  $\{v_1 \dots v_n\}$  so  $(v_{n-1}, v_n) \in E$ . Choose in order with Pr.  $\frac{1}{2}$ .

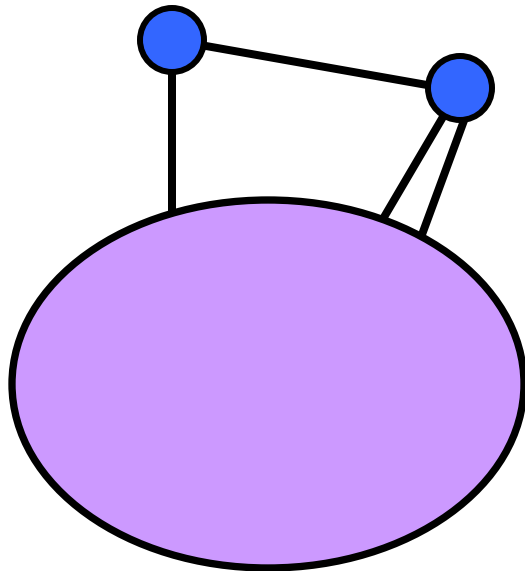


*even case*



# Verification

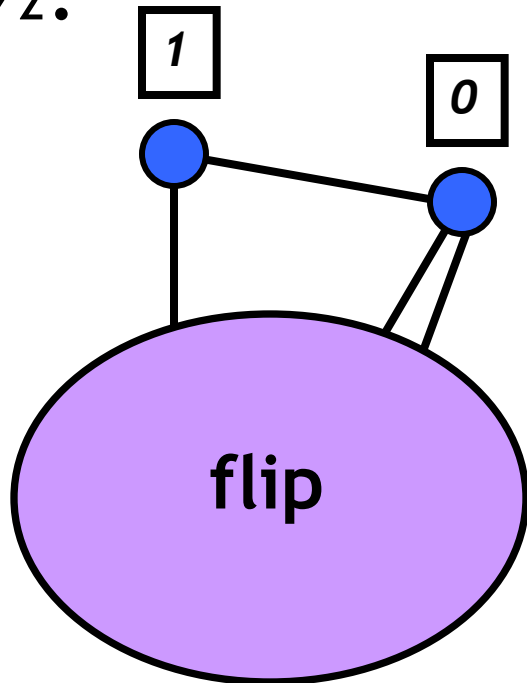
- **Pf of Lemma:** Order the vertices  $\{v_1 \dots v_n\}$  so  $(v_{n-1}, v_n) \in E$ . Choose in order with Pr.  $\frac{1}{2}$ .



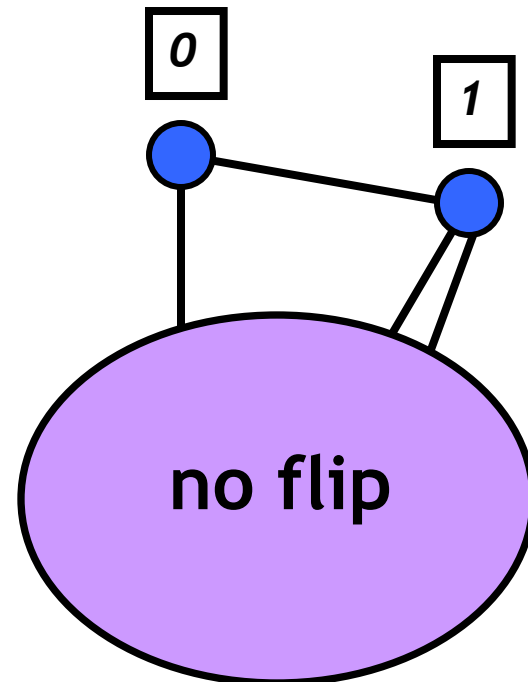
*odd case*

# Verification

- **Pf of Lemma:** Order the vertices  $\{v_1 \dots v_n\}$  so  $(v_{n-1}, v_n) \in E$ . Choose in order with Pr.  $\frac{1}{2}$ .

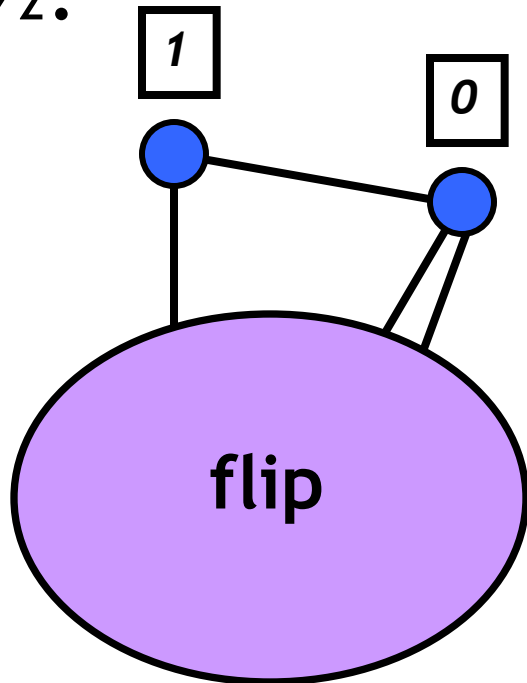


*odd case*

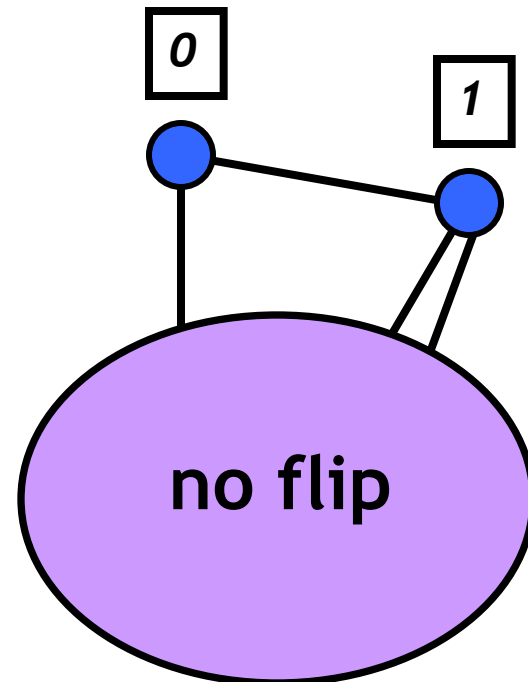


# Verification

- **Pf of Lemma:** Order the vertices  $\{v_1 \dots v_n\}$  so  $(v_{n-1}, v_n) \in E$ . Choose in order with Pr.  $\frac{1}{2}$ .



*odd case*



# Fingerprinting

- Freivalds:
  - If  $A \neq B$  then with prob.  $\frac{1}{2}$   $Ax \neq Bx$  for random  $x$ .
  - Corollary: If  $A \neq B$  then with prob.  $\frac{1}{4}$   $x^T A y \neq x^T B y$  for random independent  $x, y$ .
- This result:
  - If  $A^T + A \neq B^T + B$  then  $x^T A x \neq x^T B x$  with prob.  $\frac{1}{4}$ .
  - Weighted, directed graphs.

# Future Work

- $O(\log n)$  gap for EC/partition
- Costs can depend on query size
- Combinations of queries
- Other graph properties
- Restricted classes of graphs (e.g. trees)

# Acknowledgements

- Dana Angluin, Pradipta Mitra, Daniel Spielman for discussions
- Referee for simplifying proof of EC/Partition